Episode Book

EPISODE #6685

How Old Database Limits Become Password Policies

PUBLISHED	19 November 2026 at 17:06 UTC
DURATION	31:30
LISTEN	myweirdprompts.com/episode/legacy-password-rules-nist-iso-59772/
HOSTS	Corn, Herman, Dorothy, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
SOURCES	24 cited by the research (listed at the end)
BOOK GENERATED	19 November 2026 at 22:18 UTC

SYNOPSIS

NIST's 2025 revision 4 of 800-63B explicitly forbids mandatory symbol rules and recommends 15-character minimums, yet websites still silently truncate passwords and reject symbols. The culprit is a compliance gap: ISO 27001 audits only check that a password policy exists, not whether it works. Meanwhile, 2004-era database column limits get laundered into official rules, and some sites even measure percent-encoded bytes instead of characters. Plus, a 1861 toast sandwich recipe from Mrs Beeton's Book of Household Management.

TRANSCRIPT



Corn

Every password form on earth promises you security and then quietly negotiates. Daniel wrote in with the negotiation he keeps losing. He uses a password manager, generates strong unique passwords, does everything right — and keeps colliding with organisations, very often Israeli ones, whose rules are written down nowhere. Two specific failure modes. The site has a maximum length it never mentions, so the manager generates twenty characters and the site either rejects it or, worse, silently truncates it. And symbol validation: the policy demands a symbol, then rejects half the symbols on offer. He suspects Hebrew sites suffer more because of Unicode across two scripts. His questions: why does this happen, is there a standard anyone could just adopt, and can a site publish its rules in machine-readable form so password managers can actually follow them? Part venting, part genuine question. So let's start with the obvious one — why doesn't everyone just follow one sensible standard?



Herman

And the answer to that is more annoying than the problem, I think. Because a sensible standard exists. It's been sitting there for over a year in final form. The issue is who it binds and who it doesn't.



Corn

Go on then.



Herman

Daniel's instinct is to reach for ISO, because ISO is what organisations get audited against. And ISO — this is the part I love — ISO 27001, the 2022 revision, handles passwords through a control numbered 5.17, "Authentication information." It says you must have a managed process for allocating and controlling authentication information. That's it. There's a line I keep coming back to from one of the compliance analyses: none of the ISO controls contains a number. No minimum length. No composition formula. No rotation schedule. Bitwarden's own ISO guidance says it flat out — organisations define their authentication requirements themselves and justify them through risk assessment.



Corn

So ISO tells you to have a policy the way a doctor tells you to have a diet.



Herman

Without ever weighing you, yes. And the auditor's job is to check that the policy exists and is applied — not that it's any good. A 2010-era policy of eight characters, upper, lower, digit, symbol, rotate every ninety days, sails through an ISO audit. That template is still what most auditors expect, because auditors have long memories and nobody ever got failed for being boring.



Corn

But hold on — how does that actually play out in an audit room? I'm imagining the auditor showing up. What does the conversation look like?



Herman

Roughly this. The auditor asks to see the password policy. Someone produces a PDF. The auditor checks that the PDF exists, that it says something plausible, and that the helpdesk can confirm new accounts follow it. Whether the policy is good for anything is not on the checklist. If anything, a weird policy is a liability in an audit — if you're the only bank in the country with no rotation requirement, even though NIST now says rotation is bad, you spend the meeting explaining yourself. The safe move, auditorially, is to look like 2010.



Corn

So the incentive gradient points backwards. The more standardised your paperwork, the smoother your audit, and the standardised paperwork froze over a decade ago.



Herman

Nobody at the audit is asking "is this policy protecting anyone." They're asking "does this policy match the policy-shaped hole in my form." And a policy-shaped hole has no opinions about Unicode.



Corn

Meanwhile the actual security guidance has reversed on every one of those points.



Herman

Every single one. NIST's revision 4 of 800-63B, final form July 2025 — and this is the document Daniel is imagining when he says "minimum twelve characters, any characters allowed." He's actually underselling it. Fifteen characters minimum if the password is the sole authentication factor. Eight only when it's part of multi-factor. Verifiers should permit a maximum length of at least sixty-four characters. Accept all printable ASCII plus spaces, and it recommends accepting Unicode, counting each code point as one character.



Corn

Fifteen as the floor, sixty-four as the ceiling of the ceiling.



Herman

The floor for your maximum, technically. A site can allow more. And the composition question — the "must contain a symbol" rule — that went from SHOULD NOT to SHALL NOT. That's a hardening. Revision 3 said don't do it; revision 4 says you are forbidden to do it. Every site demanding one symbol and one digit with a twelve-character cap is doing the thing the primary standards body now prohibits.



Corn

Let me make sure I've got the force of that language right, because it sounds like legal boilerplate until you say it out loud. In standards-speak, SHALL NOT is as hard as it gets?



Herman

As hard as it gets for that document, yes. SHOULD NOT is strong advice — you'd better have a documented reason to ignore it. SHALL NOT is a prohibition. So the trajectory is: in revision 3, composition rules were a bad idea you could justify. In revision 4, they're something you must not do. And yet walk down any login page on earth and count how many are still doing it. The standards body moved from "please stop" to "stop" and the web, collectively, did not notice.



Corn

So the sensible standard exists, and it says the opposite of what most sites do. Which raises the question Daniel actually asked: can anyone just invent their own rules?



Herman

Formally, yes. There's no international standard constraining what a site may require of a password. ISO is silent, NIST binds US federal systems and is advice for everyone else. An Israeli bank or a health fund here is fully compliant with ISO 27001 while enforcing rules NIST explicitly forbids. Both things are true at once. That's the structural answer to Daniel's "why does this happen" — the standard that governs the organisation says nothing about the thing that breaks, and the standard that says something doesn't govern the organisation.



Corn

The mess isn't an accident. It's a vacancy.



Herman

It's a vacancy with an audit certificate on the door.



Corn

I want to pull at the legacy side too, because I don't think pure bureaucracy explains a twelve-character maximum. Nobody chooses twelve for policy reasons. Twelve smells like a database column.



Herman

You're right, and this is where it gets archaeological. Fixed-width columns from the mid-2000s, mainframe password fields, RACF on the old Israeli enterprise systems, Active Directory's historical 127-character ceiling. The limit is real — it's a physical constraint of something sitting under seventeen layers of stuff nobody wants to touch. What's missing is any admission that it exists. The organisation inherits a twelve-character column from 2004, someone writes "maximum 12 characters" into a policy document on an intranet, and that intranet page is the closest thing to documentation the password manager will ever see. Which is to say, nothing.



Corn

Walk me through the laundering, because I think it happens in stages. Nobody sits down and decides to lie about this.



Herman

Nobody decides anything, that's the point. Stage one, 2004: a developer defines a column, twelve characters, generous for the era, ships it. Stage two, maybe 2011: the system gets rebuilt around the old database, because you don't touch the old database, and the new form inherits the limit without anyone naming it. Stage three, someone in compliance needs to write down the password rules, asks the developers what the rules are, and the developers say — honestly — "twelve characters." Not "the column is twelve wide." "Twelve characters." And that goes in the policy document as a decision, retroactively, with an author and a date, as though someone weighed the options.



Corn

And from that day forward it has a policy number and a document owner and it can survive any audit, because it's now officially a choice.



Herman

A choice made by a schema in 2004 and cosigned by everyone who never read it since. The technical limit gets laundered into a policy, and the policy gets laundered into folklore.



Corn

And folklore can't be parsed by software. That's the bit that should sting anyone building a password manager — the most important input your generator needs is transmitted by oral tradition.



Herman

There's a nice test for how deep the folklore goes. Try asking an organisation why the limit is twelve. Not what the limit is — why that number. If you ever get an answer at all, it'll be some version of "security," delivered with the confidence of someone who has never once been asked. The number itself is unexplainable because it was never decided. It was found, like a rock in a field, and built around.



Corn

Well, that's the password business for now. Dorothy, are you there? You had a toast fact for us, didn't you?

D

Dorothy

Hello dear, it's your mother. Dorothy's History of Toast.

D

Dorothy

Mrs Beeton's Book of Household Management, from eighteen sixty-one, has a recipe for a toast sandwich. One slice of toast between two slices of bread, recommended for invalids.

D

Dorothy

So there you are. Toast, and then bread again, and then toast, and it's meant to help. I suppose it mostly shows things don't work out.

D

Dorothy

Anyway. I've got the kettle on and your father's socks to sort. Be good, sweetheart.



Herman

Mum, that's the finest thing I've heard all week.



Corn

Medicine was a simpler grift back then.



Herman

I'll take it over the intranet policy document, honestly. Where were we — legacy limits. The thing about the truncation failure, the one Daniel flagged as the worse of his two, is that it doesn't even announce itself. You generate a twenty-character password, the site takes the first twelve, and now the password in your manager is not the password on the server. You only find out at login, and the error message says "incorrect password," which is technically true and completely useless.



Corn

That one's insidious because the failure is deferred. The form accepts it. Everything looks fine. The bomb goes off next Tuesday.



Herman

And think about when it goes off, too. You set the password once, on signup, when you're in a hurry. The login failure arrives later — a different day, a different device maybe, a moment when you actually need the thing. So the user's mental model isn't "the site truncated my password." The user's mental model is "the password manager got it wrong," or "I'm typing something wrong," or "this service is broken." Nobody's mental model includes silent truncation, because nothing in the visible world suggests it.



Corn

And the user's next move, nine times out of ten, is to fall back to something they can type from memory.



Herman

Which is the bitter punchline. The tool built to let you use strong passwords everywhere fails in a way that pushes you back to a weak one you can retype. The truncation bug doesn't just break the password — it breaks the user's trust in the correct behaviour.



Corn

And Daniel's Hebrew hypothesis — I went looking for whether anyone has actually studied this, and there is exactly one paper, from 2012, by Joseph Bonneau and Rubin Xu, presented at a workshop at Cambridge. The title is the thesis: "Of contraseñas, , and " — Spanish, Hebrew, and Chinese for "passwords." They tested sites and found IMDB and Walmart enforcing length limits by counting bytes instead of characters. Both on ISO-8859-1, both apparently not stripping the percent-encoding from the submitted password before measuring it.



Corn

Wait, say that again slowly. They were measuring the password after it had been encoded for transmission, not the password itself?



Herman

That's what the evidence pointed to, yes. When a form submits a password, the characters get percent-encoded for the wire — anything outside plain ASCII becomes percent signs and hex digits. If your length check runs on that encoded string, you're not counting characters, you're counting transport packaging. It's like a post office charging postage by the size of the envelope the letter was photocopied onto.



Corn

The 2012 versions of those sites. Large engineering organisations.



Herman

Which is the point — this isn't a "badly built website" problem, it's a layers problem. And the mechanism is worse than it sounds. Percent-encoding a non-ASCII character expands it. Their worked example is the Chinese character for "love": six bytes on the wire in one encoding, nine in UTF-8, fourteen under ISO 8859-1, because that one turns it into an HTML entity first. One character. A twelve-character password with a few non-ASCII characters in it can arrive at the server as over a hundred bytes.



Corn

Twelve characters in, a hundred bytes out. That's not a password anymore, that's freight.



Herman

And any byte limit — a legacy column, a form-length check — trips on it. For Hebrew specifically: the script sits above the ASCII range entirely, so every single Hebrew character is multi-byte in UTF-8 and a percent-encoded triplet on the wire. There is no cheap Hebrew. An English site with a byte limit can limp along for years because most of its users stay under the limit by accident. A Hebrew site hits it with every third login attempt. Daniel's suspicion is a well-founded hypothesis with a mechanism behind it, not just annoyance.



Corn

And it's not just the password field itself, surely. If the whole site's plumbing assumes one byte equals one character, the password is just where you notice.



Herman

It's where you notice because it's the only field where the mismatch survives to bite you later. A mangled address in a delivery field gets seen and fixed by a human. A mangled password gets hashed, stored, and waits. The password is the one piece of input whose only test is a future comparison against itself.



Corn

And the two-script case is where it stops being a length problem and becomes a text-direction problem.



Herman

Bidi, yes. If you mix Latin letters, Hebrew letters, and symbols in one field, you inherit all the bidirectional text rules, plus the normalisation problem — two strings that both display identically can hash differently. The Bonneau and Xu data on this is small but delightful. They got hold of a leak from a Hebrew site called Wonder-Tree, about a thousand credentials from 2011. Roughly 2.5 percent of the passwords used non-ASCII characters, mostly Hebrew. One of them contained a bidirectional control character — almost certainly from a copy-paste. That person does not know their password contains an invisible character.



Corn

Their password contains a character that reorders the rest of the password.



Herman

Which is sitting in the field, changing the display order of everything around it, and hashing along with it. And the deeper finding — most users, across all four languages they looked at, actively avoid non-ASCII even when it's allowed. They transliterate, they remap their keyboards to produce nonsense ASCII, or they fall back to all-numeric passwords. The authors' concern was that this reduces guessing resistance for non-English speakers, which I find quietly sad — the people dodging the encoding bugs end up with weaker passwords than the bugs would have caused.



Corn

Punished for routing around the problem. And it's not even a conscious workaround, is it? Nobody sits down and decides "I'll avoid Hebrew for technical reasons." They just have one bad experience — a password that works on one device and not another — and quietly retreat.



Herman

Learned helplessness, encoded one user at a time. And the cruel part is that the retreat is invisible to the site. The site never sees the passwords that weren't tried. There's no error log entry for "user gave up and used their cat's name in English."



Corn

There was one more gem in that paper. Browser disagreement. A page in a Chinese encoding with no code point for the Spanish enye — Internet Explorer would send it one way, Chrome and Firefox a different way, and the authors note that neither choice was clearly endorsed by any standard. Your password manager and your browser could disagree about what your password is, at the byte level, invisibly.



Herman

And note what that means for the whole premise of a password manager. The manager's job is to store the password and reproduce it exactly. But "exactly" isn't fully determined by the manager — it's determined by the manager, the browser, the page encoding, and the server's decoding, all agreeing. Any link in that chain can reinterpret, and the reinterpretation is silent.



Corn

We did an episode a while back on what your messaging app isn't telling you — the gap between the promise on the tin and the machinery underneath. Same genus. The password form says "enter your password." The machinery underneath is arguing about bytes.



Herman

Which leads us to the part of Daniel's question that has a real answer. Machine-readable publication of the rules. It exists. It has existed since 2018. It is not standardised. There's a proposal for the HTML standard, issue number 3518 in the WHATWG repository, opened on the first of March 2018 by an Apple WebKit engineer. As of March this year it is still



Herman (continued)

open, labelled "needs implementer interest." Eight years.



Corn

An HTML attribute stuck in committee for eight years. The web moves fast except where it matters.



Herman

The syntax is lovely, actually. You put a passwordrules attribute on the password input, and it's semicolon-delimited pairs — required upper, required lower, required digit, max-consecutive two — alongside the normal minlength and maxlength attributes. Character classes: upper, lower, digit, special, ascii-printable, unicode. And if the attribute is absent, the default is allowed ascii-printable. The proposal's own motivation is Daniel's complaint almost word for word. It says the best known solution today is a generator rule that matches many sites, plus — this is their phrase — a curated list of per-site quirks for sites with unusual requirements. A better solution would be for the website to express its requirements in machine-readable form.



Corn

Stop on that phrase. "Curated list of per-site quirks." That's the industry admitting the rules are invisible.



Herman

It's the workaround with a name. And it exists — it's not hypothetical. Apple maintains a repository called Password Manager Resources, and 1Password uses it to identify a website's rules, plus a file mapping domains that share one account system. So when your manager generates a sensible password for a site that never published anything, it's guessing from a hand-maintained database that someone, a human being, populated by hitting that site's form and recording what happened.



Corn

How does that even work in practice, though? Who's adding the entries? Is there a queue somewhere?



Herman

It's a public repository, so it's pull requests — an engineer at one of the companies, or an motivated outside contributor, encounters a site whose rules break generation, works out what the rules actually are by experimentation, and files the finding. Then someone reviews it, because of course a malicious or merely wrong entry in that list is a security problem in itself. You're trusting strangers to describe other strangers' login forms accurately. It's Wikipedia for constraints.



Corn

And the coverage question follows immediately. The big American sites, sure — millions of users, someone has hit every quirk and filed it. What about a health fund in Haifa?



Herman

That's exactly the gap. The quirks list is proportional to traffic and to the language of the people maintaining it. A site that's huge in one country and invisible in the GitHub anglosphere can sit there for years with wrong rules and no entry, and every one of its users pays the toll individually. Daniel isn't hitting the gap by bad luck. He lives in the gap.



Corn

Mechanised folklore. We've reinvented the village elder.



Herman

Bitwarden's in too — their SDK parses the attribute, and they actually delegate the parsing to a parser crate that 1Password maintains. And they merged a test in June this year for a create-account flow with passwordrules. So you've got Apple originating, 1Password consuming the quirks list, Bitwarden building against the attribute — three password managers, one proposal, zero standard.



Corn

Which means a site that publishes the attribute today is publishing in a dialect.



Herman

A de-facto Apple, 1Password, Bitwarden dialect. Not part of the HTML standard, because the proposal never landed. And most sites — and I'd bet most Israeli ones, since a small site nobody has added to the quirks repository is invisible to every manager — publish nothing at all. Not the attribute, not the quirks entry, nothing. Your generator is flying blind over Jerusalem.



Corn

There's a detail in Apple's implementation I want to ask you about, because it smells like a fight. Apple ignores the rules if the length is under twelve or if fewer than two character classes are allowed.



Herman

The anti-degradation guard. If a site's published rules would force a password too weak or too narrow, Safari just… declines to comply. Silently, presumably. The site asked for garbage and Apple pretends not to have heard.



Corn

The password manager overruling the site it's trying to serve.



Herman

Which is either paternalism or mercy depending on your angle. And here's the paradox sitting underneath the whole attribute — the proposal itself warns that each property-value pair reduces the entropy of the generated password. Every rule you publish makes the password weaker, because every rule is information an attacker can use. Allowed unicode and nothing else gives the strongest generated passwords. Five required clauses gives the weakest.



Corn

Hold on. The sites with the most paranoid, most detailed, most specified policies — those are producing the weakest generated passwords.



Herman

That's the arithmetic, yes. The paranoia is subtractive. And it loops back to NIST beautifully — the composition rules that NIST now forbids are exactly the entropy the proposal is warning you about. Every party to this story knows the rules make things worse. The rules persist anyway, because an auditor in 2010 wrote them down and nobody has been paid to unwrite them.



Corn

It's worth spelling out why a rule leaks, because it's not obvious. If I know your password is exactly twelve characters with one symbol and one digit, what have I actually gained as an attacker?



Herman

You've shrunk the space I have to search. Every constraint cuts the set of possible passwords, and I, the attacker, get to apply every cut you published. A twelve-character password drawn from everything is a bigger haystack than a twelve-character password with mandated structure, because the mandated structure tells me where the needle isn't. Multiply that across millions of users on one site and the attacker isn't attacking one password — they're attacking the shape of all of them at once.



Corn

So the site that publishes the most rules is doing the attacker's preprocessing for free.



Herman

Compliance as a gift to the enemy. An attribute that punishes you for using it correctly. Herman, this is my favourite piece of web infrastructure since we found out how TCP treats everyone equally badly.



Hilbert

Rane rotary trim, seventeen dollars a channel, if anyone's counting levels. Corn, that meter on your second word there — I'll smooth it in the edit. Anyway. The tilde. There's a system I use, weekly, that requires a symbol and will not accept the ampersand. Took me four attempts to find that out, because the error just says invalid character. Not which character. The policy document says must contain a symbol. It does not say which symbols. Nobody at the organisation could tell me. I worked it out myself in the end — the



Hilbert (continued)

password goes through a URL as a query string on the backend, unencoded, so the ampersand gets read as a separator. Everything after it goes to a different field. I tested every symbol on the keyboard in order until one was accepted. The tilde came ninth.



Herman

You systematically brute-forced your own password form.



Hilbert

Somebody had to. There's a note in my manager entry for that site. It says use the tilde. That note is the only documentation that system has ever had.



Herman

That's better documentation than most of these sites produce, honestly.



Hilbert

It's correct, it's short, and it doesn't lie. Can't say that for the policy document. Herman, you said the standards body forbade the symbol rule. Nobody at that place has read a standards body since the fax machine. Fourteen fields on that form and the password was the only one that argued with me. Corn, I'm cutting the pause you left before your last question, it reads as doubt. The tilde holds. It has held for years.



Corn

I'd trust the tilde over the policy anyway. One of them has been tested.



Herman

That's the thing I keep circling, actually. The workaround is more reliable than the rule. The quirks repository, your note, the tilde — the working state of the art is individual humans recording what actually happened to them, one site at a time.



Corn

And there's a durability question hiding in that, isn't there? Your note says use the tilde. What happens the day the backend gets fixed?



Herman

Nothing, at first — the tilde keeps working, because a fix only widens what's accepted. The note goes stale in the harmless direction. But flip it around for the quirks repository: a site changes its rules, nobody updates the entry, and now the manager is generating to a policy that no longer exists. Folklore has no expiry date either. The village elder's knowledge rots as quietly as the intranet page did.



Hilbert

Mine hasn't rotted. I check. Every few months I try the ampersand again, quietly, to see if they've fixed it. They have not. Four years running.



Corn

Science, in its purest form.



Herman

Replication with negative results, unpublished, in a notes field.



Corn

Cutting-room floor before we close — one detail from the Bonneau and Xu paper we didn't get to. In 2012 they found nine major sites with explicit written policies barring non-ASCII passwords entirely. Google, Microsoft Live, Yahoo, Amazon, among others. Most of those bans lived in client-side JavaScript. Whatever your feelings about a ban, at least it was written down — more than most of the sites we've been talking about manage.



Herman

Nine explicit bans, and in the same study seven sites — Facebook, Wikipedia, Twitter — happily accepted passwords of over a hundred non-ASCII characters. It was always possible. Some just decided to. And the fact that the bans were client-side JavaScript is its own little confession — it means the server probably would have accepted the characters fine. The wall was decorative.



Corn

Which is almost the definition of the whole problem. Rules enforced at the edge, for reasons nobody at the centre remembers, against a backend that never asked for them.



Herman

So here's where I land on Daniel's question. The standard he wants exists, and it's stricter than he guessed — but it binds nobody who's inconveniencing him, and the standard that does bind them is silent on purpose. The machine-readable answer exists too, and it's been sitting in an open proposal for eight years, fluent in exactly three implementations. The question worth holding onto isn't whether passwordrules will ever become a standard — it's whether any site would publish its rules if it did. Publishing means admitting the rules, and the rules, half the time, are a database column from 2004 wearing a policy hat.



Corn

Until that changes, the curated list of per-site quirks is the actual solution, which means the problem is solved by strangers with notes. I find that both bleak and weirdly reassuring.



Herman

Producer credit to Hilbert Flumingtop, who documents the tilde so the rest of us don't have to. If you enjoyed this one, try episode thirty-five, The Privacy Gap; episode sixteen, On Deepfakes, SynthID, And AI Watermarking; and episode nine, Benchmarking Custom ASR Tools - Beyond The WER. This has been My Weird Prompts — if today's episode vindicated your own doomed symbols, leave us a review, and you can find the whole archive at my weird prompts dot com. We'll be back soon.



Corn

See you tomorrow, hopefully with fewer characters rejected.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **NIST, Digital Identity Guidelines: Authentication and Authenticator Management, SP 800-63B-4, July 2025 (approved 2025-05-30). [primary]**
   https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63B-4.pdf

2. **NIST SP 800-63B web edition, dated 2025-08-26. [primary]**
   https://pages.nist.gov/800-63-4/sp800-63b.html

3. **WHATWG HTML issue #3518, Proposal: HTML passwordrules attribute, opened 2018-03-01, updated 2026-03-18, state: open. [primary]**
   https://github.com/whatwg/html/issues/3518

4. **Apple Developer Documentation, Customizing Password AutoFill rules, retrieved 2026-11-19. [primary]**
   https://developer.apple.com/documentation/security/customizing-password-autofill-rules

5. **1Password developer docs, Design your website to work best with 1Password, retrieved 2026-11-19. [primary]**
   https://www.1password.dev/web/compatible-website-design

6. **Bitwarden SDK API docs, `bitwarden_generators::passwordrules`, retrieved 2026-11-19. [primary]**
   https://sdk-api-docs.bitwarden.com/bitwarden_generators/passwordrules/index.html

7. **Bitwarden SDK, `parse_password_rules`, retrieved 2026-11-19. [primary]**
   https://sdk-api-docs.bitwarden.com/bitwarden_generators/passwordrules/fn.parse_password_rules.html

8. **Bitwarden PR #449, Add create account flow with passwordrules, merged 2026-06-24.**
   https://github.com/bitwarden/test-the-web/pull/449

9. **Joseph Bonneau & Rubin Xu, Of contraseñas, , and : Character encoding issues for web passwords, W2SP 2012, Cambridge Computer Laboratory.**
   https://www.ieee-security.org/TC/W2SP/2012/papers/w2sp12-final7.pdf

10. **Bitwarden, ISO 27001: Password policy requirements and management, retrieved 2026-11-19.**
    https://bitwarden.com/resources/iso-27001-password-policy-requirements/

11. **Compyl, Password Policy Requirements 2026: ISO 27001, NIST, PCI, retrieved 2026-11-19.**
    https://compyl.com/blog/what-to-know-about-the-iso-27001-password-policy/

12. **ISO/IEC 27002:2022 control 5.17 summary, retrieved 2026-11-19.**
    https://compliance.manzill.com/iso-iec-27002-2022/5.17/organizational-controls/authentication-information/

13. **Philippine privacy guidance summarising ISO/IEC 27002 password requirements, 2023.**
    https://privacy.gov.ph/wp-content/uploads/2023/10/Guidelines-on-Passwords.pdf

14. **Netwrix, NIST password guidelines: SP 800-63B explained, retrieved 2026-11-19.**
https://netwrix.com/en/resources/blog/nist-password-guidelines/

15. **CybersecKyle, Responding to NIST's 2025 Password Standard Update (SP 800-63B-4), published 2025-11-10.**
https://www.kylereddoch.me/blog/responding-to-nists-2025-password-standard-update-sp-800-63b-4/

16. **Junify, How long should a password be?, retrieved 2026-11-19.**
https://www.junify.com/guides/shared-accounts/password-length-recommendations

17. **Passwork, NIST password guidelines 2026: What SP 800-63B-4 requires, retrieved 2026-11-19.**
https://passwork.pro/blog/nist-password-guidelines-2026-what-sp-800-63b-4-requires/

18. **Wikipedia, Password policy, last edited 2026-10-22.**
https://en.wikipedia.org/wiki/Password_policy

19. **Wikipedia, Password strength, last edited 2026-11-18.**
https://en.wikipedia.org/wiki/Password_strength

20. **Mozilla bug 451155, password manager failure on IDN sites with characters above U+0100.**
https://bugzilla.mozilla.org/show_bug.cgi?id=451155

21. **Stack Overflow, Should I support Unicode in passwords?**
https://stackoverflow.com/questions/1797777/should-i-support-unicode-in-passwords

22. **Unicode FYI, Unicode in Passwords and Authentication, retrieved 2026-11-19.**
https://unicodefyi.com/guide/unicode-authentication/

23. **Scott Brady, Perfecting the password field with the HTML passwordrules attribute, retrieved 2026-11-19.**
https://www.scottbrady.io/authentication/perfecting-the-password-field-with-the-html-passwordrules-attribute

24. **Lewis Boon, Use Apple's Password Rules to help Password Managers, 2021-11-16.**
https://lewisboon.co.uk/blog/2021-11-16-apple-password-rules/